

# INF0323 — RAPPEL

## LMD - ALGÈBRE RELATIONNELLE

Nadia Tahiri, Ph. D.  
Professeure adjointe  
Université de Sherbrooke

*Nadia.Tahiri@USherbrooke.ca*

© Anis Boubaker (2018), Robert Godin (2012)



# ALGÈBRE RELATIONNELLE

- Les langage de requête SQL permet d'exprimer des requêtes sur des ensembles (relations) qui correspondent à des fonctions algébriques:
  - La projection
  - La sélection
  - Le produit cartésien
  - La jointure
  - La division
  - ...
- Chaque opération produit une relation (possible de composer)
- Permet de représenter des plans d'exécution de requêtes

# L'INSTRUCTION SELECT

- Syntaxe de la commande SELECT:

```
SELECT [DISTINCT] {listeExpressions | *}  
FROM table [AS nomTable] [,table [AS nomTable]]...  
[spécificationJointure]...  
[WHERE conditionSQL]  
[GROUP BY nomColonne,[nomColonne]...]  
[HAVING conditionSQL ]  
[ORDER BY nomColonne [ASC | DESC] [, nomColonne [ASC | DESC]...]
```

# RELATION / TABLE

- Tous les étudiants de la tables étudiants

## Etudiant

```
SELECT *
FROM Etudiant
```

code_perm	nom	prenom	date_naiss
SNOJ1982102101	Snow	Jon	1982-10-21
STAA1987071509	Stark	Arya	1987-01-10
TARD1979071502	Targaryen	Daenerys	1982-10-21
GRET1981091501	Greyjoy	Theon	1981-09-15
LANT1970061203	Lannister	Tyrion	1970-06-12

# PROJECTION D'UNE TABLE


(  $\Pi$  )

- Le nom et le prénom de tous les étudiants de la table étudiant

$\Pi_{\text{nom, prénom}}(\text{Etudiant})$

```
SELECT nom, prenom
FROM Etudiant
```

code_perm	nom	prenom	date_naiss
SNOJ1982102101	Snow	Jon	1982-10-21
STAA1987071509	Stark	Arya	1987-01-10
TARD1979071502	Targaryen	Daenerys	1982-10-21
GRET1981091501	Greyjoy	Theon	1981-09-15
LANT1970061203	Lannister	Tyrion	1970-06-12



nom	prenom
Snow	Jon
Stark	Arya
Targaryen	Daenerys
Greyjoy	Theon
Lannister	Tyrion

# PROJECTION D'UNE TABLE

( $\Pi$ )

- La date de naissance de tous les étudiants de la table étudiant

$\Pi_{\text{date\_naiss}}(\text{Etudiant})$

```
SELECT date_naiss
FROM Etudiant
```

code_perm	nom	prenom	date_naiss
SNOJ1982102101	Snow	Jon	1982-10-21
STAA1987071509	Stark	Arya	1987-01-10
TARD1979071502	Targaryen	Daenerys	1982-10-21
GRET1981091501	Greyjoy	Theon	1981-09-15
LANT1970061203	Lannister	Tyrion	1970-06-12



date_naiss
1982-10-21
1987-01-10
1982-10-21
1981-09-15
1970-06-12

Multi-ensemble!

# PROJECTION D'UNE TABLE


( $\Pi$ )

- La date de naissance de tous les étudiants de la table étudiant

$\Pi_{\text{date\_naiss}}(\text{Etudiant})$

```
SELECT DISTINCT date_naiss
FROM Etudiant
```

code_perm	nom	prenom	date_naiss
SNOJ1982102101	Snow	Jon	1982-10-21
STAA1987071509	Stark	Arya	1987-01-10
TARD1979071502	Targaryen	Daenerys	1982-10-21
GRET1981091501	Greyjoy	Theon	1981-09-15
LANT1970061203	Lannister	Tyrion	1970-06-12



date_naiss
1982-10-21
1987-01-10
1981-09-15
1970-06-12

# LA SÉLECTION

( $\sigma$ )

- Les produits dont le prix excède 5.00\$

$\sigma_{\text{prix} > 5.00}(\text{Produit})$

```
SELECT *
FROM Produit
WHERE prix > 5.00
```

id_prd	nom	prix
10	Baguette régulière	3.00
11	Baguette à l'ancienne	5.50
12	Ficelle	2.75
13	Pain de campagne	9.95
14	Toutière aux oignons	15.25



id_prd	nom	prix
11	Baguette à l'ancienne	5.50
13	Pain de campagne	9.95
14	Toutière aux oignons	15.25



# COMBINER LES OPÉRATIONS

- Le nom et le prix des produits dont le prix excède 5.00\$

$\Pi_{\text{nom, prix}}(\sigma_{\text{prix} > 5.00}(\text{Produit}))$

```
SELECT nom, prix
FROM Produit
WHERE prix > 5.00
```

id_prd	nom	prix
10	Baguette régulière	3.00
11	Baguette à l'ancienne	5.50
12	Ficelle	2.75
13	Pain de campagne	9.95
14	Toutière aux oignons	15.25

$\Pi_{\text{nom, prix}}$   
 $\sigma_{\text{prix} > 5.00}(\text{Produit})$



nom	prix
Baguette à l'ancienne	5.50
Pain de campagne	9.95
Toutière aux oignons	15.25

# TABLE VIRTUELLES - VUES

- Le résultat d'une requête est une table virtuelle. (ou logique)
- Le résultat d'une table virtuelle peut-être stocké sous la forme d'une vue
- Syntaxe (partielle) pour la création de vues:

```
CREATE OR REPLACE VIEW Nom_Vue  
AS requeteSQL;
```

```
CREATE OR REPLACE VIEW Produits_Dispendieux  
AS  
SELECT nom, prix  
FROM Produit  
WHERE prix>5.00
```


# TABLE VIRTUELLES - VUES

- Une vue ne contient pas de données – Tire ses données depuis les tables de base;
- Une vue peut-être utilisée comme une table
  - ✓ Il est même possible d’y insérer des données si on respecte certaines conditions !

**Table produits**

id_prd	nom	prix
10	Baguette réguliere	3.00
11	Baguette à l'ancienne	5.50
12	Ficelle	2.75
13	Pain de campagne	9.95
14	Toutière aux oignons	15.25

```
SELECT *  
FROM Produits_Dispendieux
```



nom	prix
Baguette à l'ancienne	5.50
Pain de campagne	9.95
Toutière aux oignons	15.25

# L'INSTRUCTION SELECT

- Syntaxe de la commande SELECT :

```
SELECT [DISTINCT] {listeExpressions | *}  
FROM table [AS nomTable] [,table [AS nomTable]]...  
[spécificationJointure]...  
[WHERE conditionSQL]  
[GROUP BY nomColonne,[nomColonne]...]  
[HAVING conditionSQL ]  
[ORDER BY nomColonne [ASC | DESC] [, nomColonne [ASC | DESC]...]
```


	conditionSQL
[NOT] {	
expression {=   <   >   <=   >=   <>}	expression
expression BETWEEN expression AND expression	
expression {IS NULL   IS NOT NULL}	
expression {IN   NOT IN} listeConstantes	
expression {LIKE   NOT LIKE} patron	
} [{AND   OR} <b>conditionSQL</b> ]...	

# CONDITION SQL - BETWEEN

- Les produits dont le prix se situe entre 2.00\$ et 5.00\$

```
SELECT nom, prix  
FROM Produit  
WHERE prix BETWEEN 2.00 AND 5.00
```

id_prd	nom	prix
10	Baguette réguliere	3.00
11	Baguette à l'ancienne	5.50
12	Ficelle	2.75
13	Pain de campagne	9.95
14	Toutière aux oignons	15.25




nom	prix
Baguette réguliere	3.00
Ficelle	2.75

# CONDITION SQL – [NOT] IN

- Les produits ayant pour identifiants 11, 13 et 14

```
SELECT *  
FROM Produit  
WHERE id_prd IN (11,13,14)
```

id_prd	nom	prix
10	Baguette régulière	3.00
11	Baguette à l'ancienne	5.50
12	Ficelle	2.75
13	Pain de campagne	9.95
14	Toutière aux oignons	15.25




id_prd	nom	prix
11	Baguette à l'ancienne	5.50
13	Pain de campagne	9.95
14	Toutière aux oignons	15.25

# CONDITION SQL - IS [NOT] NULL

- Les étudiants dont on ne connaît pas la date de naissance

```
SELECT nom, prenom  
FROM Etudiant  
WHERE date_naiss IS NULL
```

code_perm	nom	prenom	date_naiss
SNOJ1982102101	Snow	Jon	1982-10-21
STAA1987071509	Stark	Arya	NULL
TARD1979071502	Targaryen	Daenerys	1982-10-21
GRET1981091501	Greyjoy	Theon	NULL
LANT1970061203	Lannister	Tyrion	1970-06-12



nom	prenom
Stark	Arya
Greyjoy	Theon

# CONDITION SQL - BETWEEN

- Les produits dont le prix se situe entre 3.00\$ et 5.00\$

```
SELECT nom, prix  
FROM Produit  
WHERE prix BETWEEN 3.00 AND 5.00
```

id_prd	nom	prix
10	Baguette réguliere	3.00
11	Baguette à l'ancienne	5.50
12	Ficelle	2.75
13	Pain de campagne	9.95
14	Toutière aux oignons	15.25



nom	prix
Baguette réguliere	3.00
Ficelle	2.75



# CONDITION SQL – [NOT] LIKE

## (AVEC CHAINES DE CARACTÈRES)

- Les codes permanents des étudiants donc le prénom commence, par T, comprend au moins 3 lettres et se termine par N

```
SELECT code_perm  
FROM Etudiant  
WHERE prenom LIKE 'T_%n'
```

% : zéro ou plusieurs caractères  
\_ : exactement un caractère

code_perm	nom	prenom	date_naiss
SNOJ1982102101	Snow	Jon	1982-10-21
STAA1987071509	Stark	Arya	NULL
TARD1979071502	Targaryen	Daenerys	1982-10-21
GRET1981091501	Greyjoy	Theon	NULL
LANT1970061203	Lannister	Tyrion	1970-06-12

**code\_perm**

GRET1981091501

LANT1970061203

# LE RENOMMAGE

( $\rho$ )

- Les noms de produits et leurs prix. Les produits doivent s'appeler des "produit\_boulangier"

$\rho$  nom/produit\_boulangier (  $\sigma_{\text{prix} > 5.00}(\text{Produit})$  )

```
SELECT nom AS produit_boulangier, prix  
FROM Produit
```

id_prd	nom	prix
10	Baguette réguliere	3.00
11	Baguette à l'ancienne	5.50
12	Ficelle	2.75
13	Pain de campagne	9.95
14	Toutière aux oignons	15.25



produit_boulangier	prix
Baguette réguliere	3.00
Baguette à l'ancienne	5.50
Ficelle	2.75
Pain de campagne	9.95
Toutière aux oignons	15.25

# LE RENOMMAGE D'UNE RELATION (TABLE)

- Il est possible de renommer une table pour :
  - Faciliter l'écriture des conditions / jointures
  - Lever l'ambiguïté lors d'auto-jointures

```
SELECT nom, prix  
FROM Produit P  
WHERE P.id_prd BETWEEN 11 AND 13;
```

- Le nouveau nom peut même être utilisé dans la liste de colonnes:

```
SELECT P.nom, P.prix  
FROM Produit P  
WHERE P.id_prd BETWEEN 11 AND 13;
```